DERWENT-ACC-NO: 1998-236084

DERWENT-WEEK: 199821

TITLE: Debugging system used in program development of computer - in which addition unit provided in first computer couples debug information compressed by compression unit and executable program which is to be debugged

PRIORITY-DATA: 1996JP-0229697 (August 30, 1996)

PATENT-FAMILY:

PUB-NO	PUB-DATE	LANGUAGE	PAGES	MAIN-IPC
JP 10074152 A	March 17, 1998	N/A	011	G06F 011/28

INT-CL (IPC): G06F011/28

ABSTRACTED-PUB-NO: JP 10074152A

BASIC-ABSTRACT:

The system includes an addition unit (108) to couple the debug information compressed by a compression unit (111) and an executable program (105) in a first computer (101). The coupled program compressed debug information is input into the memory (114) of an second computer (102) by an input unit (113).

The second computer executes the program. A reading unit (119) of a third computer (103) reads out the compressed debug information and the program from the memory of second computer. An expansion unit (121) expands the read compressed debug information. A debugger (122) debugs the executed program.

ADVANTAGE - Avoids need for matching program and debug information. Enables using memory effectively. Provides debug information on computer for reading out debug information.

---------- KWIC ---------

Basic Abstract Text - ABTX (1):

The system includes an addition unit (108) to couple the debug information compressed by a compression unit (111) and an executable program (105) in a first computer (101). The coupled program compressed debug information is input into the memory (114) of an second computer (102) by an input unit (113).

Basic Abstract Text - ABTX (2):

The second computer executes the program. A reading unit (119) of a third computer (103) reads out the compressed debug information and the program from the memory of second computer. An expansion unit (121) expands the read

compressed debug information. A debugger (122) debugs the executed program.

Basic Abstract Text - ABTX (3):
ADVANTAGE - Avoids need for matching program and debug information. Enables using memory effectively. Provides debug information on computer for reading out debug information.

Derwent Accession Number - NRAN (1):
1998-236084

Title - TIX (1):
Debugging system used in program development of computer - in which addition unit provided in first computer couples debug information compressed by compression unit and executable program which is to be debugged

Standard Title Terms - TTX (1):
DEBUG SYSTEM PROGRAM DEVELOP COMPUTER ADD UNIT FIRST COMPUTER COUPLE DEBUG INFORMATION COMPRESS COMPRESS UNIT EXECUTE PROGRAM DEBUG

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平10－55289

(43)公開日　平成10年(1998)２月24日

(51)Int.Cl.[6]	識別記号	庁内整理番号	ＦＩ		技術表示箇所
Ｇ０６Ｆ　11/28	３１０		Ｇ０６Ｆ　11/28	３１０Ａ	
11/34			11/34	Ｎ	

審査請求　未請求　請求項の数４　ＯＬ　（全 11 頁）

(21)出願番号　特願平8－212522

(22)出願日　平成８年(1996)８月12日

(71)出願人　000006013
三菱電機株式会社
東京都千代田区丸の内二丁目２番３号

(72)発明者　佐藤　尚和
東京都千代田区丸の内二丁目２番３号　三菱電機株式会社内

(72)発明者　吉田　豊彦
東京都千代田区丸の内二丁目２番３号　三菱電機株式会社内

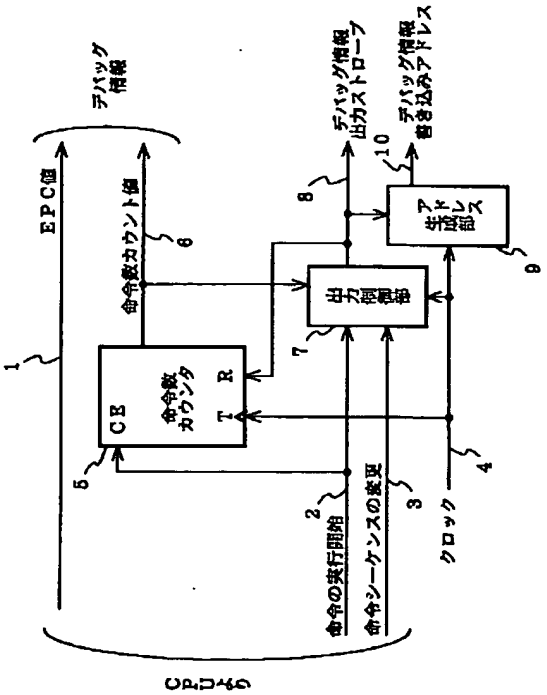
(74)代理人　弁理士　田澤　博昭　（外２名）

(54)【発明の名称】　デバッグ情報生成回路、およびそれを用いた情報処理装置

(57)【要約】

【課題】　新しく命令の実行が開始される都度、そのＥＰＣ値を出力してＰＣ値トレース用メモリに格納しておく必要があり、大容量のメモリが必要となるという課題があった。

【解決手段】　実行された命令のＰＣ値が直前の命令のＰＣ値と連続していない場合、あるいは実行された命令の数を計数している命令数カウンタ５による命令数カウント値６が所定の比較値２５に達した場合に、当該命令数カウンタ５の命令数カウント値６とそのときの命令のＥＰＣ値１とによって生成されるデバッグ情報１０８の出力を制御するとともに、デバッグ情報１０８の出力終了後、命令数カウンタ５の初期化を行う出力制御部７を設けたものである。



1

【特許請求の範囲】

【請求項1】　実行された命令の数を計数する命令数カウンタと、

ある時刻に実行された命令がその直前に実行された命令のプログラムカウンタ値をインクリメントすることによって呼び出されたものではなかった場合に、前記ある時刻に実行された命令のプログラムカウンタ値と前記命令数カウンタによる命令数カウント値からなるデバッグ情報の出力を制御し、当該デバッグ情報が出力された後に前記命令数カウンタを初期化する出力制御部とを備えたデバッグ情報生成回路。

【請求項2】　出力制御部が、命令数カウンタによる命令数カウント値を所定の比較値と比較する比較器を有し、

前記比較器の比較結果が真である場合にもデバッグ情報の出力を制御して、当該デバッグ情報が出力された後に前記命令数カウンタを初期化するものであることを特徴とする請求項1記載のデバッグ情報生成回路。

【請求項3】　比較器にて命令数カウンタによる命令数カウント値と比較される所定の比較値として、前記命令数カウンタのとり得る最大値を設定したことを特徴とする請求項2記載のデバッグ情報生成回路。

【請求項4】　請求項1から請求項3のうちのいずれか1項記載のデバッグ情報生成回路を備えた情報処理装置において、

前記デバッグ情報生成回路からデバッグ情報が格納される記憶手段への前記デバッグ情報の出力を抑止する機能、

および前記デバッグ情報の前記記憶手段への出力と、当該情報処理装置自身に与えられる命令の実行によって引き起こされる前記記憶手段の読み出しの切り替えを行う機能を備えて、

前記デバッグ情報の出力が抑止されている期間にのみ、当該情報処理装置自身に与えられる命令の実行によって引き起こされる前記記憶手段へのアクセスを受け入れる制御切替手段を備えたことを特徴とする情報処理装置。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】この発明は、プログラムのデバッグを行うに際して、実行した命令とその順序を復元するためにPC値をトレースするためのデバッグ情報を生成・出力するデバッグ情報生成回路、および当該デバッグ情報生成回路を用いた情報処理装置に関するものである。

【0002】

【従来の技術】マイクロプロセッサなどの情報処理装置においてプログラムのデバッグを行う場合に、実行された命令とその実行順序を復元すること、すなわち実行された命令のプログラムカウンタ値（以下プログラムカウンタをPC、プログラムカウンタ値をPC値という）を

2

時間軸に沿って復元することはきわめて重要なことである。処理がパイプライン化されていない情報処理装置などでは、命令フェッチの様子をモニタすることにより上記目的を達成することができた。しかしながら、パイプラインマシンにおいては、フェッチされた命令が必ずしも実行されるとは限らないため、命令フェッチの様子をモニタしていたのでは上記目的を達成することができない。そのため、通常のパイプラインプロセッサでは、PC相対の分岐命令実行等のために、現在実行中の命令のPC値（以降EPC値という）を保持しており、新しく命令の実行を開始する度にEPC値を出力して、これをPC値トレース用メモリに格納しておくことにより、実行された命令のPC値を復元することを可能としている。

【0003】なお、このような従来のデバッグ情報生成回路に関連した記載のある文献としては、例えば、特開昭62-200439号公報、特開昭52-77550号公報、特開昭62-200435号公報などがある。

【0004】

【発明が解決しようとする課題】従来のデバッグ情報生成回路は以上のように構成されているので、プログラムのデバッグを行うために、新しく命令の実行が開始される都度、そのEPC値を出力してそれをPC値トレース用メモリに格納しておく必要があるが、実用的なプログラムでは実行される命令ステップは膨大な数となり、そのため、極めて大きな容量のメモリが必要となる。しかしながら、高速で動作する情報処理装置の実行サイクル毎にデータの書き込みが可能な程度のメモリ装置は高価なものであり、大きな容量のメモリを実装することは経済的な負担が著しく増大するという課題があった。

【0005】この発明は上記のような課題を解決するためになされたもので、プログラムデバッグのためのトレースデータを圧縮して、それを格納するためのメモリの容量を削減し、経済的に有利なデバッグ情報生成回路、およびそれを用いた情報処理装置を得ることを目的とする。

【0006】

【課題を解決するための手段】請求項1記載の発明に係るデバッグ情報生成回路は、実行された命令のPC値が直前の命令のPC値と連続していない場合に、実行された命令の数を計数している命令数カウンタの命令数カウント値とそのときの命令のEPC値とによって生成されるデバッグ情報の出力を制御し、出力終了後、命令数カウンタの初期化を行う出力制御部を設けたものである。

【0007】請求項2記載の発明に係るデバッグ情報生成回路は、デバッグ情報の出力制御を、命令数カウンタによる命令数カウント値が所定の比較値に達した場合にも実行するようにしたものである。

【0008】請求項3記載の発明に係るデバッグ情報生成回路は、命令数カウンタのとり得る最大値を、当該命

令数カウンタの命令数カウント値と比較される比較値として用いたものである。

【0009】請求項4記載の発明に係る情報処理装置は、当該情報処理装置自身に与えられる命令の実行によって引き起こされる記憶手段へのアクセスを、デバッグ情報生成回路からのデバッグ情報の出力が抑止されている期間にのみ受け入れる制御切替手段を有するものである。

【0010】

【発明の実施の形態】以下、この発明の実施の一形態を説明する。

実施の形態1．図1はこの発明の実施の形態1によるデバッグ情報生成回路の構成を示すブロック図であり、このデバッグ情報生成回路は、マイクロプロセッサなどによる情報処理装置の図示を省略した中央演算処理装置（以下ＣＰＵという）に付加されて、ＰＣ値のトレースを行うためのデバッグ情報を出力するものである。図において、1はＣＰＵで現在実行している命令のＰＣ値であるＥＰＣ値であり、2はＣＰＵが新しい命令の実行を開始したことを示す命令の実行開始信号である。3は分岐命令の実行や割り込みなどの割り込み例外（以下ＥＩＴという）の処理のためにＥＩＴベクタへのジャンプ等が生じた場合にＣＰＵより出力されて、ＣＰＵで実行されているプログラムにおいて命令シーケンスが変更されたことを示す、命令シーケンスの変更信号であり、この命令シーケンスの変更信号3が出力された場合、次に実行される命令は新しいシーケンスの先頭となる。4は処理動作の同期をとるためのクロック信号である。

【0011】また、5は実行された命令の数を計数する命令数カウンタ、6はこの命令数カウンタ5によって計数された命令数カウント値であり、この実施の形態1ではこの命令数カウンタ5として8ビットカウンタを仮定して説明を行う。この命令数カウンタ5内のＣＥはＣＰＵからの命令の実行開始信号2が入力されるカウントイネーブル端子、Ｔはクロック信号4が入力されるクロック入力端子であり、カウントイネーブル端子ＣＥにイネーブルが与えられている場合に、クロック端子Ｔに入力されたクロック信号4の立ち上がりエッジに同期して命令数カウント値6を更新する。なお、カウントイネーブル端子ＣＥがディセーブルの場合には、クロック信号4が与えられても命令数カウント値6を更新しない。また、Ｒはこの命令数カウンタ5の初期化を行うためのリセット端子であり、クロック信号4の立ち上がりに同期して命令数カウント値6を初期化する。ここで、カウントイネーブル端子ＣＥとリセット端子Ｒとに同時に信号が与えられた場合には、リセット端子Ｒの制御を優先する。なお、ここでは図示していないが、このデバッグ情報生成回路が用いられている情報処理装置のシステム全体に対するリセット信号が与えられた場合にも、この命令数カウンタ5は初期化される。この命令数カウンタ5

から出力される命令数カウント値6はＣＰＵからのＥＰＣ値1とともに、デバッグ情報として出力される。

【0012】7はＣＰＵより命令の実行開始信号2と命令シーケンスの変更信号3が与えられた場合、あるいは命令数カウンタ5からの命令数カウント値6が所定の値に達した場合に、ＥＰＣ値1と命令数カウント値6とによるデバッグ情報を出力するためのデバッグ情報出力ストローブを生成する出力制御部であり、8はこの出力制御部7より出力されるデバッグ情報出力ストローブである。なお、このデバッグ情報出力ストローブ8は、デバッグ情報の出力後に命令数カウンタ5を初期化するため、そのリセット端子Ｒにも入力されている。9はこの出力制御部7からのデバッグ情報出力ストローブ8を受け取って、デバッグ情報を出力する度にデバッグ情報書き込みアドレスのインクリメントを行うアドレス生成部であり、10はこのアドレス生成部9より出力されるデバッグ情報書き込みアドレスである。

【0013】次に動作について説明する。ここで、この実施の形態1によるデバッグ情報生成回路において、ＥＰＣ値1と命令数カウント値6とによるデバッグ情報が出力されるのは、次の2つの場合である。

【0014】すなわち、その一つは、分岐命令の実行や割り込み等のＥＩＴ処理の発生などによって命令シーケンスが変更になった場合であり、シーケンス変更後、最初の命令が実行されたときにデバッグ情報を出力する。この時のデバッグ情報は、シーケンス変更後にどの命令に実行が移ったか、前回のシーケンス変更から今回のシーケンス変更までに何個の命令を実行したかを示したものである。最近接のシーケンス変更の間ではＰＣ値のインクリメントとともに命令実行が進められたことが保証されているので、実行された命令の個数が判明すれば、この間に実行された命令のＰＣ値を復元することができる。

【0015】また、他の一つは、命令数カウンタ5が所定の比較値に達したときである。この場合、現在実行中の命令が、あたかもシーケンス変更後の最初の命令であるかのごとく扱い、デバッグ情報を出力する。この場合にも、隣接する2回のデバッグ情報の間では、ＰＣ値のインクリメントとともに命令実行が進められたことが保証されているので、実行された命令の個数より実行された命令のＰＣ値を復元することができる。この処理は、シーケンス変更が極めて少ない場合に命令数カウンタ5がオーバーフローして、正しい実行状態が復元できなくなるのを防止するためのものである。

【0016】図2（ａ）～図4（ｂ）は、ＥＰＣ値1、命令の実行開始信号2、命令シーケンスの変更信号3の様子をいくつかの例について示した説明図である。なお、ここでは、以下に示す5段のパイプラインマシンを仮定している。

第1段（Ｉステージ）：命令フェッチを行う

第2段（Dステージ）：命令デコードを行う
第3段（Eステージ）：命令の実行を行う
第4段（Mステージ）：メモリアクセスを行う
第5段（Wステージ）：ロードオペランドの書き込みを行う

【0017】また、ここで例示した命令は次のような処理を行うものとする。

ADD　ra，rb　レジスタraの値とレジスタrbの値を加算して結果をレジスタraに格納する。1サイクルで実行を完了する。

SUB　ra，rb　レジスタraの値からレジスタrbの値を減算して結果をレジスタraに格納する。1サイクルで実行を完了する。

DIV　ra，rb　レジスタraの値を被除数、レジスタrbの値を除数として除算を行い、結果をレジスタraに格納する。実行に2サイクル以上を要する。

LD　ra，rb　レジスタrbの値をアドレスとしてメモリからデータのロードを行い、ロード結果をレジスタraに格納する。

ST　ra，rb　レジスタrbの値をアドレスとして、レジスタraの値をメモリに書き込む。

JMP　dstn　本命令の実行完了後、dstnで示される命令を実行する。1サイクルで実行を完了する。

【0018】以下、図2（a）～図4（b）に示すそれぞれの場合における、EPC値1、命令の実行開始信号2、および命令シーケンスの変更信号3の様子について順番に説明する。図2（a）は通常の加算命令が3つ連続した場合を示したものである。この場合、毎サイクル新しい加算命令が実行され、EPC値1は各加算命令の実行サイクル（Eステージ）においてそれぞれのPC値となり、それぞれの加算命令の実行サイクル毎に命令の実行開始信号2が出力される。なお、この場合、命令シーケンスの変更信号3は出力されない。

【0019】図2（b）は1命令の実行に複数サイクルを要する除算命令がある場合を示したものである。この場合、この除算命令に続く各加算命令は、当該除算命令の全ての実行サイクルが終了するまで実行待ちの状態となる。したがって、除算命令の最初の実行サイクルにおいてEPC値1は当該除算命令のPC値となって、この除算命令の全実行サイクルが終了するまでその値が保持され、その後、各加算命令の実行サイクル毎にそれぞれのPC値となる。一方、命令の実行開始信号2は除算命令の最初の実行サイクルで出力されて2サイクル目以降の実行サイクルでは出力されず、この除算命令の実行サイクルが完了すると、各加算命令の実行サイクル毎に出力される。なお、この場合も命令シーケンスの変更信号3は出力されない。

【0020】図3（a）はロード命令等のメモリアクセスを行う命令が連続し、かつ最初のメモリアクセスが1サイクルで終了しなかった場合を示したものである。この場合、図中にPC値が@（N＋1）で示される2番目のロード命令のメモリアクセスサイクル（Mステージ）は、@（N）で示される最初のロード命令の全てのメモリアクセスサイクルが終了するまで実行待ちの状態となり、それに続く加算命令も2番目のロード命令のメモリアクセスサイクルが開始されるまで実行待ちの状態となる。したがって、命令の実行開始信号2は最初のロード命令の実行サイクル、および2番目のロード命令の最初の実行サイクルで出力された後、加算命令の実行サイクルが開始されるまで出力されない。また、EPC値1は2番目のロード命令の実行サイクルにて当該命令のPC値となると、次の加算命令の実行サイクルが開始されるまでその値が保持される。なお、この場合も命令シーケンスの変更信号3は出力されない。

【0021】図3（b）は図中にPC値が@（N＋1）で示される加算命令が、@（N）で示される完了していないロード命令のデスティネーションレジスタr1を参照しようとして実行を待たされている場合を示したものである。この場合、@（N）のロード命令の実行サイクルで命令の実行開始信号2が出力され、EPC値1が当該ロード命令のPC値となる。ここで、このロード命令に続く@（N＋1）の加算命令は、参照するデスティネーションレジスタr1にデータがロードされるまで実行待ち状態となっているので、この実行を待たされている間は命令の実行開始信号2は出力されず、EPC値1もそのままの値を保持する。なお、この場合も命令シーケンスの変更信号3は出力されない。

【0022】図3（c）はノーディレイドの分岐命令が実行された場合について示したものである。この場合、図中にPC値が@（N）で示される分岐命令の実行サイクルが終了すると、それに続く@（N＋1）、@（N＋2）で示される加算命令の命令デコードサイクル（Dステージ）や命令フェッチサイクル（Iステージ）は行われず、@（M）で示される書き込み命令の命令フェッチ、さらには@（M＋1）で示される減算命令の命令フェッチが行われる。なお、この分岐命令の実行サイクルの開始から終了までの間、命令シーケンスの変更信号3および命令の実行開始信号2が出力される。一方、この分岐命令の実行サイクルの開始時点でEPC値1が当該ロード命令のPC値となり、ストア命令の実行サイクルが開始されるまでその値が保持される。

【0023】図4（a）は分岐によるパイプラインフラッシュをなくした特殊な分岐が実行された場合について示したものであり、図5に示す機構によって実現されるものである。図5において、11はPCであり、12は分岐先命令アドレスをこのPC11に設定するための分岐先命令アドレスバスである。13は分岐先命令を指定する分岐先命令レジスタ、14は分岐元命令を指定する分岐元命令レジスタであり、15はこの分岐元命令レジ

スタ１４の値とＰＣ１１のＰＣ値とを比較する比較器である。１６は分岐が生じない場合にＰＣ値をインクリメントするインクリメンタであり、１７はＰＣ１１より命令フェッチを行う際の命令のアドレスが出力される命令アドレスバスである。

【００２４】なお、この図５には図示していないが、ここで示す特殊な分岐の実行をイネーブルとするか否かを指定する制御ビットがプロセッサ・ステータス・ワード（以下ＰＳＷという）内に備えられているものとする。この制御ビットが真のときに比較器１５の比較結果が真であれば、分岐先命令レジスタ１３の値を分岐先命令アドレスバス１２に出力し、ＰＣ１１は現在の命令フェッチが完了した後、上記分岐先命令アドレスバス１２の値を取り込む。すなわち、命令フェッチの段階で実行すべき命令のシーケンス変更を行うものである。図４（ａ）においては、ＥＰＣ値１が@（Ｎ）の加算命令が分岐元命令レジスタ１４と一致した場合を示したものである。特殊な分岐が実行された場合には、対象となった分岐元命令に特定のフラグをセットしておけば、このフラグを参照することによってＥＰＣ値１が@（Ｎ）の加算命令の実行を開始したときに、命令シーケンスの変更信号３を出力することができる。

【００２５】図４（ｂ）は割り込みやリセットなど（以下割り込みという）が生じた場合について示したものであり、この場合、当該割り込みは図中にＰＣ値が@（Ｎ）で示される加算命令の実行サイクルにおいて検出されたものとする。割り込みが検出されると、最初の加算命令に続く@（Ｎ＋１）および@（Ｎ＋２）で示される加算命令の命令デコードサイクルや命令フェッチサイクルは行われず、それらが行われるサイクルにおいてその割り込みのＥＩＴ処理が実行される。したがって、最初の加算命令の実行サイクルの開始時点でＥＰＣ値１が当該命令のＰＣ値となり、命令の実行開始信号２が出力される。なお、上記割り込みのＥＩＴ処理では、ＥＩＴ要求を検出した次のサイクルでＥＩＴハンドラの先頭番地（ＥＩＴベクタ）の生成を行い、さらに次のサイクルでＥＩＴベクタをＰＣにセットする。このＥＩＴベクタのＰＣへのセットと同時に命令シーケンスの変更信号３が出力される。その後、ＰＣにセットされたＥＩＴベクタによって、@（Ｍ）の書き込み命令の命令フェッチが行われる。なお、この書き込み命令の実行サイクルが開始されるまで、命令の実行開始信号２は出力されず、ＥＰＣ値１は最初の加算命令のＰＣ値をそのまま保持し続ける。

【００２６】次に、このような命令の実行開始信号２および命令シーケンスの変更信号３を受けた出力制御部７の動作について説明する。ここで、図６は、この出力制御部７の構成例を示したブロック図である。図において、２１は１ビットのレジスタ、２２はこのレジスタ２１の蓄積信号、２３は論理ゲートであり、論理ゲート２３の出力はレジスタ２１に蓄えられ、論理ゲート２３はこのレジスタ２１の蓄積信号２２および命令の実行開始信号２と、命令シーケンスの変更信号３の論理演算結果をレジスタ２１に入力している。また、２４は比較器であり、２５はこの比較器２４によって命令数カウント値６が比較される所定の比較値、２６はこの比較器２４から出力される一致信号である。２７は命令の実行開始信号２に同期して、レジスタ２１の蓄積信号２２もしくは比較器２４の一致信号２６をデバッグ情報出力ストローブ８として出力する論理ゲートである。

【００２７】次に動作について説明する。命令シーケンスの変更信号３が与えられると論理ゲート２３の出力はアサートされ、それがレジスタ２１に蓄積されてその蓄積信号２２は真となる。また、レジスタ２１の蓄積信号２２が真であれば、命令の実行開始信号２が与えられていない状態で論理ゲート２３の出力はアサートされる。したがって、レジスタ２１の蓄積信号２２は一旦真になると、次のサイクル以降において命令の実行開始信号２がアサートされるまでその値を保持する。命令の実行開始信号２がアサートされると、レジスタ２１の蓄積信号２２は論理ゲート２７よりデバッグ情報出力ストローブ８として出力される。この命令の実行開始信号２のアサートにより、論理ゲート２３の出力はアサートされなくなり、レジスタ２１の蓄積信号２２は偽となる。なお、この状態は、次に命令シーケンスの変更信号３が与えられるまで保持される。

【００２８】一方、比較器２４は命令数カウンタ５からの命令数カウント値６をあらかじめ設定された所定の比較値２５と比較している。命令数カウント値６が比較値２５と等しくなると比較器２４は一致信号２６を出力し、この一致信号２６は論理ゲート２７に送られる。論理ゲート２７は命令の実行開始信号２がアサートされると、その一致信号２６をデバッグ情報出力ストローブ８として出力する。なお、この処理は命令数カウンタ５のオーバーフローによってＰＣ値を復元するために必要な情報が失われることを防ぐものである。したがって、比較器２４で命令数カウント値６と比較される比較値２５には、命令数カウンタ５の最大値を設定しておけばよい。この実施の形態１では命令数カウンタ５として８ビットカウンタを想定しているので、比較値２５には最大値２５５が設定されているものとする。このようにすることにより、命令数カウンタ５がオーバーフローする直前にデバッグ情報が出力されるようになるため、比較器２４の一致信号２６によるデバッグ情報の出力は最少限に抑えることができ、デバッグ情報格納用のメモリの容量をより削減することが可能となる。もちろん、これ以外の値を設定することも可能であり、別途用意したレジスタなどの値を参照することにより、比較値２５を変更可能とすることもできる。

【００２９】次に、命令数カウンタ５、出力制御部７お

よびアドレス生成部9による当該デバッグ情報生成回路の動作シーケンスを図7に示す。図7（a）はEPC値1が“N＋3”である命令が分岐命令であった場合を示している。この場合には、EPC値1が“N＋3”の命令が分岐命令であるので、そのとき命令の実行開始信号2とともに命令シーケンスの変更信号3もアサートされる。したがって、その分岐先の命令であるEPC値1が“M”の命令の実行開始とともに、出力制御部7より出力されるデバッグ情報出力ストローブ8がアサートされ、これによってそのときのEPC値1と命令数カウント値6によるデバッグ情報が出力される。なお、命令数カウンタ5は各サイクルにおける命令の実行開始信号2によってその命令数カウント値6をインクリメントしており、このEPC値1が“M”の命令の実行が開始されてデバッグ情報出力ストローブ8がアサートされると、命令数カウンタ5はリセットされてその命令数カウント値6が“0”となり、アドレス生成部9より出力されるデバッグ情報書き込みアドレス10がインクリメントされる。

【0030】図7（b）はEPC値1が“M＋3”である命令が分岐命令であり、かつ分岐先のEPC値1が“L”の命令もまた分岐命令であった場合について示している。この場合、EPC値1が“M＋3”の命令が分岐命令であるので、命令の実行開始信号2と命令シーケンスの変更信号3がアサートされ、その分岐先の命令であるEPC値1が“L”の命令の実行開始とともに出力制御部7より出力されるデバッグ情報出力ストローブ8がアサートされてデバッグ情報が出力される。この場合にはさらに、分岐先のEPC値1が“L”の命令も分岐命令であるので、命令の実行開始信号2と命令シーケンスの変更信号3がアサートされ、その分岐先の命令であるEPC値1が“K”の命令の実行開始とともにデバッグ情報出力ストローブ8が再度アサートされてデバッグ情報が出力される。このEPC値1が“L”である命令および“K”である命令の実行が開始されてデバッグ情報出力ストローブ8がアサートされると、アドレス生成部9からのデバッグ情報書き込みアドレス10はインクリメントされ、命令数カウンタ5からの命令数カウント値6はリセットされる。したがって、EPC値1が“K”の命令の実行開始とともに出力されるデバッグ情報においては、その命令数カウント値6に“0”が出力される。これはEPC値1が“L”の命令と“K”の命令の間では何も命令が実行されなかったことを示している。

【0031】図7（c）はEPC値1が“K”の命令を実行してから、255個以上の命令を実行してもその間になんらシーケンスの変更が無かった場合について示している。この場合、命令数カウンタ5は各サイクル毎に命令の実行開始信号2にしたがって命令数カウント値6をインクリメントしており、命令数カウント値6が“255”になると同時に出力制御部7はデバッグ情報出力ストローブ8をアサートする。このデバッグ情報出力ストローブ8のアサートによってデバッグ情報が出力されるとともに、命令数カウンタ5はリセットされてその命令数カウント値6が“0”となり、アドレス生成部9より出力されるデバッグ情報書き込みアドレス10がインクリメントされる。

【0032】次に、このようにしてデバッグ情報生成回路より出力されたデバッグ情報のメモリへの格納と、メモリに格納されたデバッグ情報から実際の命令実行順序の復元について説明する。

【0033】図8は図7（a）～（c）に示した動作シーケンスによって生成されたデバッグ情報が書き込まれたメモリの様子を示した説明図である。図示のように、図7（a）に示したEPC値1が“M”の命令が実行された時には、命令数カウント値6が“C＋3”であり、デバッグ情報書き込みアドレス10が“n”であるので、メモリのアドレス“n”のEPC値のフィールドには“M”、命令数カウント値のフィールドには“C＋3”がそれぞれ格納される。また、図7（b）に示すEPC値1が“L”の命令実行時には、命令数カウント値6が“3”、デバッグ情報書き込みアドレス10が“n＋1”、“K”の命令実行時には命令数カウント値6が“0”、デバッグ情報書き込みアドレス10が“n＋2”であるので、メモリのアドレス“n＋1”のEPC値のフィールドには“L”、命令数カウント値のフィールドには“3”が格納され、アドレス“n＋2”のEPC値のフィールドには“K”、命令数カウント値のフィールドには“0”が格納される。同様にして、図7（c）に示すEPC値1が“K＋256”の命令実行時には、命令数カウント値6が“255”、デバッグ情報書き込みアドレス10が“n＋3”であるので、メモリのアドレス“n＋3”のEPC値のフィールドには“K＋256”が、命令数カウント値のフィールドには“255”が格納される。

【0034】図9はこの図8に示したメモリの格納情報に基づいて、実行された命令のPC値を復元した結果を示す説明図である。図示のように、アドレス“n”に格納されたEPC値によって実行された命令のPC値“M”を復元し、アドレス“n＋1”に格納された命令数カウント値よりその後に3つの命令が実行されたことを知り、実行された命令のPC値“M＋1”～“M＋3”の復元を行う。次にアドレス“n＋1”に格納されたEPC値によって実行された命令のPC値“L”を復元し、アドレス“n＋2”に格納された命令数カウント値よりその後1つの命令も実行されずに命令シーケンスが変更されたことを知る。次にアドレス“n＋2”に格納されたEPC値によって実行された命令のPC値“K”を復元し、アドレス“n＋3”に格納された命令数カウント値よりその後に255命令が実行されたこと

を知り、実行された命令のPC値“K＋1”～“K＋255”の復元を行う。このようにして、隣接するアドレスに格納されているEPC値と命令数カウント値に基づいて、実行された命令のPC値を実行された順番通りに復元することが可能となる。

【0035】以上のように、この実施の形態1によれば、命令シーケンスに変更が生じた場合、および命令数カウンタ5の命令カウント値6が所定の比較値25に達した場合にのみ、EPC値1と命令数カウント値6からなるデバッグ情報を出力するようにしているので、プログラムデバッグのためのトレースデータを圧縮することができ、それを格納するためのメモリを削減することが可能となる。例えば、PCのビットサイズが32ビットで、命令数カウンタ5が8ビットの場合を想定すると、命令実行の度にEPC値1を記録するような従来の方式によると、1000命令の実行履歴を記録するためには32000ビットのメモリ容量を必要とする。しかしながら、この実施の形態1によるデバッグ情報生成回路を用いれば、2命令に1回ずつの分岐実行を仮定したとしても、40ビットのデバッグ情報を500回格納するだけであり、20000ビットのメモリ容量で事足りる。分岐実行が10回に1回の割合であれば、わずかに4000ビットの容量しか必要とせず、その効果は容易に理解できる。

【0036】また、命令数カウンタ5のオーバーフローによってPC値復元のための情報が失われることを防止することもでき、さらに、頻繁に分岐が発生するようなプログラムしか実行しない場合には、命令数カウンタ5のビット幅を小さくすることにより、さらにメモリ使用量を小さくすることも可能である。

【0037】実施の形態2．上記実施の形態1においては、プログラムのデバッグを行う際に、実行した命令とその実行順序を復元するためのPC値トレース用のデバッグ情報を生成して出力するデバッグ情報生成回路について説明したが、この実施の形態2においては、そのようなデバッグ情報生成回路を用いた情報処理装置について説明する。

【0038】ここで、図8に示したようなメモリに格納されたデータを如何にして読み出すかということに関して、この情報処理装置のシステム全体を管理するホストコンピュータにその任を負わせるという方法が考えられる。これについては図示しないが、デバッグ情報を出力するマイクロプロセッサなどによるデバッグ情報生成回路と、デバッグ情報を格納するためのメモリ装置（以下デバッグ情報メモリという）の間にバス分離装置を挿入しておき、これをホストコンピュータで制御すれば容易に実現できる。しかしながら、情報処理装置自身のCPUが自らのデバッグ情報を読み出して処理を行うか、または別のメモリ装置に値を転送することができればより容易なシステム構成が可能となる。

【0039】図10はそのような処理を実現するための情報処理装置の構成を示したブロック図であり、同一部分には図1と同一符号を付してその説明を省略する。図において、101は図1にその構成を示したデバッグ情報生成回路であり、102はデバッグを行うべきプログラムが実行されるCPU、103はデバッグ情報生成回路101からのデバッグ情報の出力を抑止する機能と、このデバッグ情報の出力と当該情報処理装置自身に与えられる命令の実行によって引き起こされる記憶手段の読み出しの切り替えを行う機能とを備え、デバッグ情報の出力が抑止されている期間のみ、当該情報処理装置自身に与えられる命令の実行によって引き起こされる記憶手段へのアクセスを受け入れる制御切替手段としてのバスインタフェースである。また、104はデバッグ情報を格納するための前記記憶手段としてのデバッグ情報メモリであり、105は通常のプログラム処理で用いられるユーザーが使用可能な外部メモリである。

【0040】106は図1に示される命令の実行開始信号2と命令シーケンスの変更信号3を統合して示した、CPU102からデバッグ情報生成回路101に送られる制御信号である。107はCPU102からバスインタフェース103に出力される、デバッグ情報の出力を行うか否かを示すデバッグ出力制御信号であり、PSW等の制御ビットの値を出力したものである。108はデバッグ情報生成回路101からバスインタフェース103に送られる、図1に示されたEPC値1と命令数カウント値6とによって生成されるデバッグ情報である。109はCPU102がデバッグ情報メモリ104にアクセスを行う際のアドレスバスであり、110は同じくCPU102がデバッグ情報メモリ104にアクセスを行う際のデータバスである。

【0041】111はバスインタフェース103とデバッグ情報メモリ104との間のデバッグ情報データバス、112は同じくデバッグ情報アドレスバスであり、113はデバッグ情報メモリ104へのデバッグ情報書き込みストローブ、114はデバッグ情報メモリ104への出力許可信号である。115はバスインタフェース103と外部メモリ105との間の外部メモリデータバス、116は同じく外部メモリアドレスバスであり、117は外部メモリ105への書き込みストローブ、118は外部メモリ105への出力許可信号である。なお、CPU102とバスインタフェース103との間、および外部メモリ105へのアクセス用の制御信号については、この実施の形態2の説明とは直接関係がないのでここではその説明を省略する。

【0042】次に動作について説明する。なお、ここでは、制御切替手段としてのバスインタフェース103の動作について、デバッグ情報メモリ104の制御に的を絞って説明する。

【0043】CPU102からバスインタフェース10

3へ送られるデバッグ出力制御信号107が真である場合には、バスインタフェース103はデバッグ情報生成回路101からのデバッグ情報108をデバッグ情報データバス111に、デバッグ情報書き込みアドレス10をデバッグ情報アドレスバス112に、デバッグ情報出力ストローブ8をデバッグ情報書き込みストローブ113にそれぞれ出力する。これによって、デバッグ情報生成回路101より出力されたデバッグ情報108が、デバッグ情報メモリ104の当該デバッグ情報生成回路101の出力するデバッグ情報書き込みアドレス10で指定されるアドレスに格納される。なお、この時にはデバッグ情報メモリ104への出力許可信号114は偽であり、CPU102によるデバッグ情報メモリ104からのデバッグ情報の読み出しは行われない。

【0044】一方、デバッグ出力制御信号107が偽であれば、バスインタフェース103はデバッグ情報メモリ104への出力許可信号114を真、デバッグ情報書き込みストローブ113を偽とする。その時、CPU102からアドレスバス109に与えられたアドレスがデバッグ情報アドレスバス112に出力されると、デバッグ情報データバス111にはこのデバッグ情報アドレスバス112によって指定されたアドレスに格納されていたデバッグ情報がデバッグ情報メモリ104より読み出されて出力される。すなわち、この状態において、デバッグ情報メモリ104はCPU102によりアクセスされる読み出し専用メモリとして動作する。ここでは、CPU102のアクセスとしてのデバッグ情報メモリ104への書き込みはサポートしないものとする。また、デバッグ出力制御信号107が真の場合にCPU102がデバッグ情報メモリ104のアクセスを要求した場合、この要求は無視される。

【0045】この図10に示した実施の形態2による情報処理装置のアドレスマップを図11に示しておく。図示の例では、論理アドレス“0000　0000”以下に内部メモリ空間が、論理アドレス“0100　0000”以下に外部メモリ105に相当する外部メモリ空間が割り当てられてユーザに開放されている。また論理アドレス“8000　0000”以下はIO空間に割り当てられ、論理アドレス“8100　0000”以下はシステムに予約され、論理アドレス“FF000000”以下はデバッグ情報メモリ104に相当するデバッグ情報メモリ空間に割り当てられて、システム予約空間となっている。図1に示したデバッグ情報生成回路の命令数カウンタ5、アドレス生成部9をそのIO空間に配置しておき、これを読み書きできるようにしておけば、デバッグ情報108の解析が容易となる。なお、このような処理は極めて常識的なことなので、ここではそれについての説明は特には行わない。

【0046】ここで、デバッグ情報108はシーケンス変更があった場合にのみデバッグ情報生成回路101より出力されるということは、何サイクルにもわたって連続してデバッグ情報108が出力されることが無いということを意味している。この場合には、図4（a）に示したPC値が@（N）の命令が別の分岐命令によって呼び出された命令であった場合にのみ、デバッグ情報生成回路101からは2サイクル連続してデバッグ情報108が出力される。なお、PC値が@（M）の命令がさらに分岐命令であったとしても、分岐先命令の実行は2サイクル以上後のことになる。すなわち、デバッグ情報108の出力は、最大の頻度でも2サイクル連続であり、またその後には必ず2サイクル以上出力されない期間が存在することになる。したがって、わずかに2エントリを備えたキューを準備すれば、命令実行の2分の1の速度でデバッグ情報108を出力することが可能となる。このようにすれば、デバッグ情報108を格納するためのメモリ装置がより低速のものでも事足りるようになるため、さらに安価なシステムを構築することが可能となる。

【0047】

【発明の効果】以上のように、請求項1記載の発明によれば、実行された命令のPC値が直前の命令のPC値と連続していない場合に、命令数カウント値とそのときの命令のEPC値とからなるデバッグ情報を生成して出力するように構成したので、プログラムデバッグのために新しく命令の実行が開始される都度そのEPC値をメモリに格納しておく必要がなくなって、プログラムデバッグのためのトレースデータを圧縮することが可能となり、それらを格納しておくためのメモリの容量を削減することができる効果がある。

【0048】請求項2記載の発明によれば、命令数カウント値が所定の比較値になった場合にもデバッグ情報を出力するように構成したので、分岐などによる命令シーケンスの変更があまりないようなプログラムをデバッグする場合でも、命令カウンタがオーバーフローして、正確なトレースデータが復元できなくなるようなことが防止できる効果がある。

【0049】請求項3記載の発明によれば、命令数カウント値が比較される所定の比較値として、命令数カウンタのとり得る最大値を用いるように構成したので、分岐などによる命令シーケンスの変更があまりないようなプログラムをデバッグする場合に、命令数カウンタがオーバーフローする直前にデバッグ情報が出力されるようになり、比較器の一致出力によるデバッグ情報の出力を最少限に抑えることができる効果がある。

【0050】請求項4記載の発明によれば、デバッグ情報生成回路からデバッグ情報の出力が抑止されている期間のみ、当該情報処理装置自身に与えられる命令の実行によって引き起こされる記憶手段へのアクセスを受け入れるように構成したので、情報処理装置自身が出力したデバッグ情報に対して、自らがアクセスすることができ

15　　16

る効果がある。

【図面の簡単な説明】

【図1】　この発明の実施の形態1によるデバッグ情報生成回路を示すブロック図である。

【図2】　この発明の実施の形態1が適用されるパイプライン形情報処理装置の動作を示すタイミングチャートである。

【図3】　この発明の実施の形態1が適用されるパイプライン形情報処理装置の動作を示すタイミングチャートである。

【図4】　この発明の実施の形態1が適用されるパイプライン形情報処理装置の動作を示すタイミングチャートである。

【図5】　この発明の実施の形態1が適用されるパイプライン形情報処理装置における特殊な分岐処理機構について示したブロック図である。

【図6】　この発明の実施の形態1における出力制御部の構成を示すブロック図である。

【図7】　この発明の実施の形態1におけるデバッグ情報生成回路の動作を示すタイミングチャートである。

【図8】　この発明の実施の形態1におけるデバッグ情報のメモリへの格納の様子を示す説明図である。

【図9】　この発明の実施の形態1におけるメモリの内容から実際の命令実行順序の復元を示す説明図である。

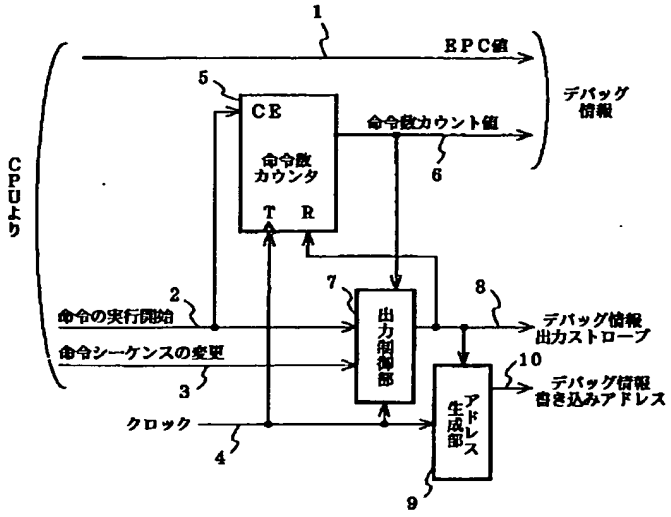
【図10】　この発明の実施の形態2による情報処理装置を示すブロック図である。

【図11】　この発明の実施の形態2におけるアドレスマップを示す説明図である。

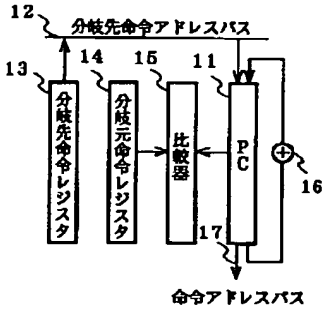
【符号の説明】

5　命令数カウンタ、6　命令数カウント値、7　出力制御部、24　比較器、25　比較値、101　デバッグ情報生成回路、103　バスインタフェース（制御切替手段）、104　デバッグ情報メモリ（記憶手段）、108　デバッグ情報。

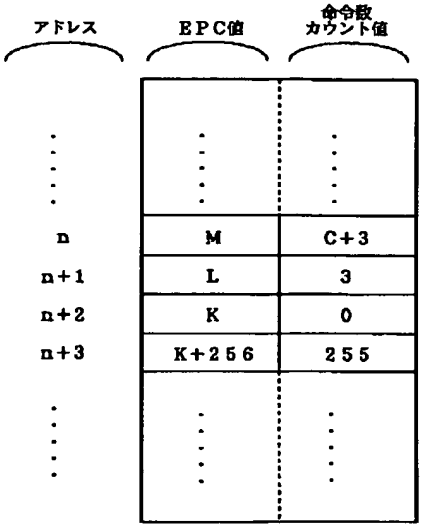
【図1】



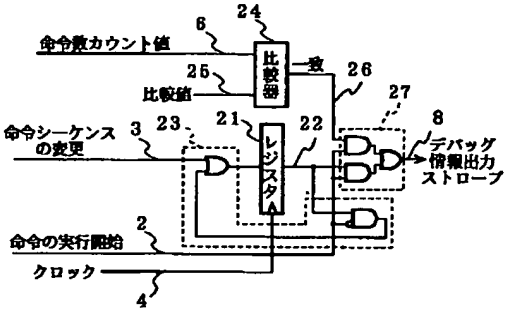
【図5】



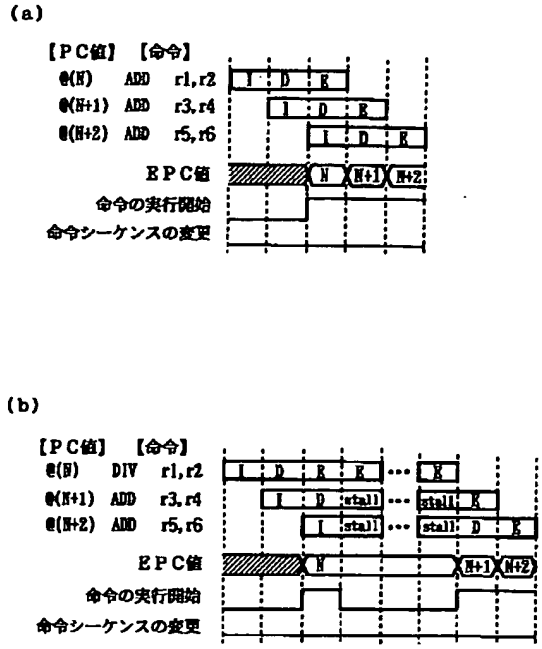
【図8】



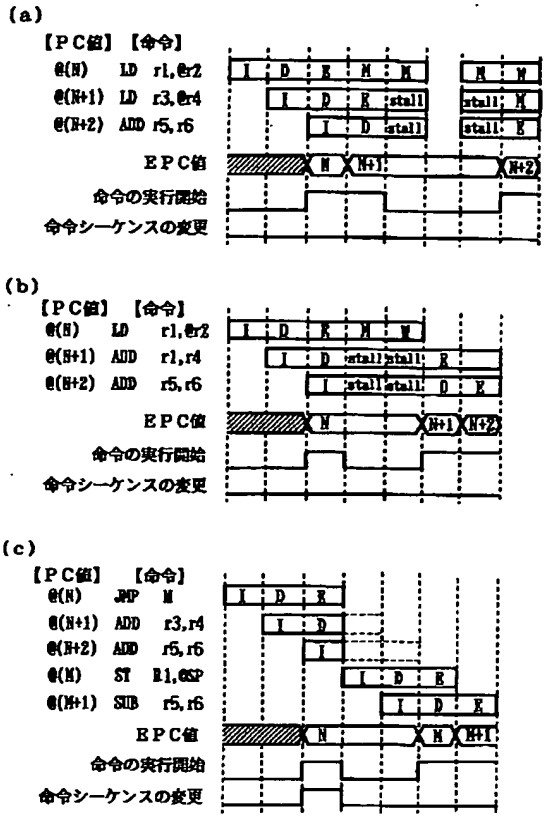
【図6】



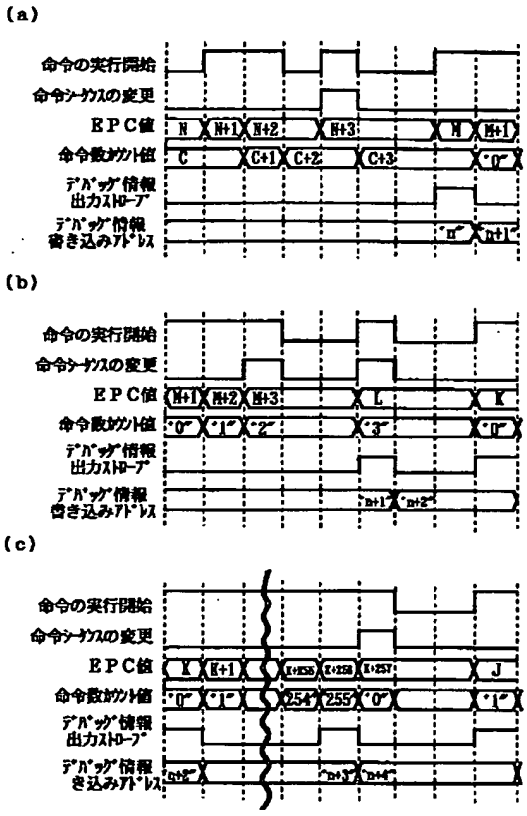
【図2】



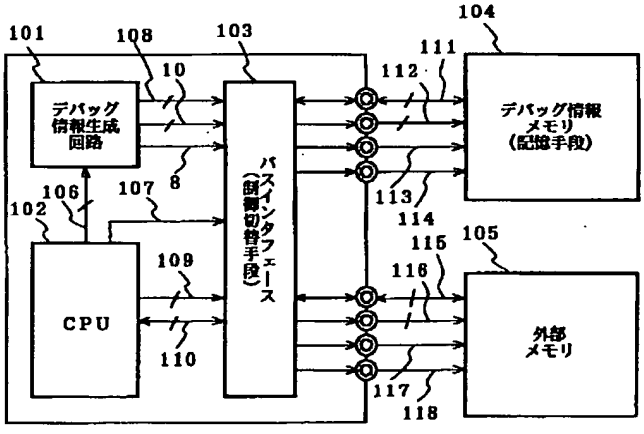
【図3】



【図7】

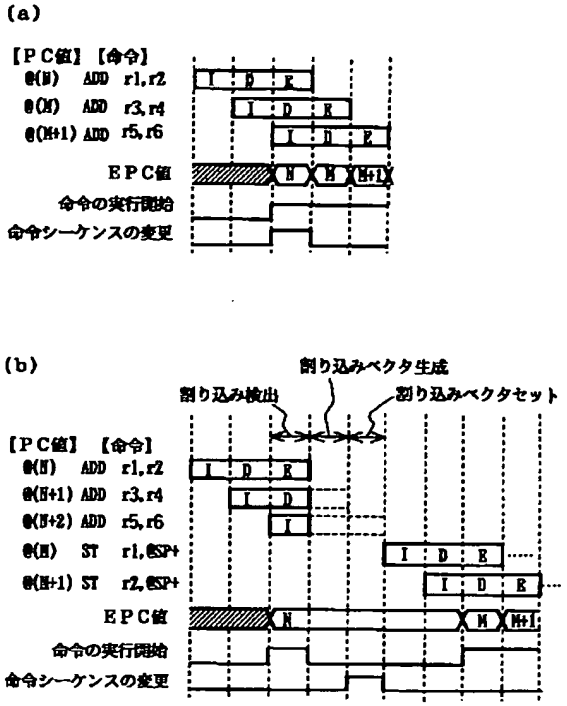


【図10】



108：デバッグ情報

【図4】



【図9】

復元した実行命令のPC値	コメント
M	← n番地のEPC値
M+1	← n+1番地の命令数カウント値より、命令@（M）の後に3命令実行されたことが判る
M+2	
M+3	
L	← n+1番地のEPC値
K	← n+2番地のEPC値 n+2番地の命令数カウント値より、命令@（K）は命令@（L）の直後に実行されたことが判る
K+1	← n+3番地の命令数カウント値より、命令@（K）の後に255命令実行されたことが判る
K+2	
K+3	
.	
.	
.	
.	
K+254	
K+255	
K+256	← n+3番地のEPC値
K+257	

【図11】